Algoritminen suunnittelu muotoilijan työkaluna

Sami Markkula
2017

Aalto-yliopisto
Taiteiden ja suunnittelun
korkeakoulu

Aalto-yliopisto, PL 11000, 00076 AALTO
www.aalto.fi

Taiteen kandidaatin opinnäytteen tiivistelmä

Tekijä Sami Markkula

Työn nimi Algoritminen suunnittelu muotoilijan työkaluna

Laitos Muotoilun laitos

Koulutusohjelma Muotoilun pääaine

Vuosi 2017 **Sivumäärä** 39 **Kieli** suomi

Tiivistelmä

Monilla suunnittelun aloilla, mutta erityisesti arkkitehtuurissa, on yleistynyt uusi algoritminen suunnitteluparadigma. Eduistaan huolimatta se ei ole vielä yleistynyt esinemuotoilun työkaluna. Tässä kandidaatintutkielmassa esitellään algoritmista suunnittelua. Lisäksi tutkielmassa käsitellään produktiota, jossa tuotettiin opetusmateriaali Aalto-yliopiston muotoilun laitoksen kurssille *Algorithms-Aided Form-Giving*.

Työssä määritellään ensin algoritmiseen muotoiluun liittyvää sanastoa ja kuvataan esimerkkien avulla muotoilutavan mahdollistamia hyötyjä. Lopuksi esitellään kandidaatintutkielman aikana tuotettu Grasshopper-opiskelumateriaali sekä pohditaan algoritmisen suunnittelun vaikutusta suunnitteluprosessiin. Tutkielmaa voidaan sellaisenaan, tai yhdessä produktio-osan kanssa, käyttää opetettaessa algoritmista suunnittelua muotoilijoille.

Tämän kandidaatintyön luettuaan lukija ymmärtää algoritmisen muotoilun käsitteitä ja tietää perusteet Grasshopper-ohjelman käytöstä. Toivon mukaan lukija innostuu algoritmisen suunnittelun tarjoamista mahdollisuuksista ja sovellutuskohteista ja alkaa pohtia algoritmisen ajattelutavan merkitystä suunnittelutyössään.

Kandidaatintyön ja opetusmateriaalin avulla muotoilija voi tutustua algoritmisen muotoilun mahdollisuuksiin ja alkaa harjoittaa algoritmista muotoilua Grasshopper 3D -ohjelmiston avulla.

Avainsanat Algoritminen muotoilu, algoritmiavusteinen muotoilu, parametrinen muotoilu, Grasshopper

Sisällys

1 Johdanto

Algoritminen suunnittelu on arkkitehtien harjoittama suunnittelun paradigma (Leach 2014). Eduistaan huolimatta se ei ole vielä yleistynyt esinemuotoilun työkaluna. Tässä kandidaatintutkielmassa esitellään algoritmista suunnittelua. Lisäksi tutkielmassa käsitellään produktiota, jossa tuotettiin opetusmateriaali Aalto-yliopiston muotoilun laitoksen kurssille *Algorithms-Aided Form-Giving*. Kandidaatintyön ja opetusmateriaalin avulla muotoilija voi tutustua algoritmisen muotoilun mahdollisuuksiin ja alkaa harjoittaa algoritmista muotoilua Grasshopper 3D -ohjelmiston avulla.

Algoritmista muotoilua voidaan käyttää muun muassa muotovariaatioiden generoimiseen, suorituskyvyn tai materiaalin optimoimiseen, massakustomointiin, käytön simulointiin, toistuvien tehtävien automatisoimiseen tai tekoälyavusteiseen muotoiluun.

Työssä määritellään ensin algoritmiseen muotoiluun liittyvää sanastoa ja kuvataan esimerkkien avulla muotoilutavan mahdollistamia hyötyjä. Lopuksi esitellään kandidaatintutkielman aikana tuotettu Grasshopper-opiskelumateriaali sekä pohditaan algoritmisen suunnittelun vaikutusta suunnitteluprosessiin. Tutkielmaa voidaan sellaisenaan, tai yhdessä produktio-osan kanssa, käyttää opetettaessa algoritmista suunnittelua muotoilijoille.

Tämän kandidaatintyön luettuaan lukija ymmärtää algoritmisen muotoilun käsitteitä ja tietää perusteet Grasshopper-ohjelman käytöstä. Toivon mukaan lukija innostuu algoritmisen suunnittelun tarjoamista mahdollisuuksista ja sovellutuskohteista sekä alkaa pohtia algoritmisen ajattelutavan merkitystä suunnittelutyössään.

2 Käsitteitä

Luvussa määritellään kandidaatintutkielmassa käytettäviä käsitteitä. Osalle termeistä on useita käyttötapoja, tässä on esitelty yhdet, tässä tutkielmassa käytettävät määritelmät.

Algoritmi on yksityiskohtainen kuvaus tai ohje siitä, miten tehtävä tai prosessi suoritetaan, ja jota seuraamalla voidaan ratkaista tietty ongelma (Tedeschi ym. 2016, 22).

Algoritminen mallinnus on tapa tuottaa algoritmisesti ennalta tiedossa oleva malli jonka tuottaminen muilla menetelmin olisi vaikeata. Varsinainen suunnittelu on jo tehty aiemmin ja algoritmisia työkaluja käytetään CAD-ohjelmien jatkeena.

Algoritminen suunnittelu, tai **algoritmiavusteinen suunnittelu,** on menetelmä, jossa algoritmeja avuksi käyttämällä etsitään ratkaisua suunnitteluongelmaan. (Tanska ja Österlund 2014, 12).

Eksplisiittinen mallinnus on tietokonemallinnusta, jossa mallinnetaan esine suoraan ilman helppoa mahdollisuutta muuttaa yksittäistä muuttujaa, muotoa tai mittasuhdetta jälkikäteen. Toisin kuin algoritminen malli, pitää eksplisiittinen malli pitää sisällään ainoastaan mallin geometrian, eikä esimerkiksi tapaa tuottaa se. Algoritminen ja parametrinen mallinnus on implisiittistä mallinnusta, ja malli seuraa määritetyistä parametreista.

Evolutiivisen algoritmin tai **Geneettisen algoritmin** avulla biologisia periaatteita käyttäen tuotetaan mallisukupolvia, joiden avulla pyritään tuottamaan yhä parempia malleja. (Tanska ja Österlund 2014, 12) Evolutiivista algoritmia varten täytyy määritellä tarkasti millä perusteella yksittäisen mallin sopivuutta arvioidaan. Tätä arviointifunktiota kutsutaan hyvyysfunktioksi *(Fitness function)*. Se voisi olla esimerkiksi astian tilavuus suhteessa painoon.

Generatiivinen suunnittelu on suunnittelua, jonka avulla luodaan, generoidaan ratkaisuja muotoiluongelmaan. Usein suunnittelulta edellytetään että laskennan tulosta käytetään määritettäessä uuden laskentakierroksen syötettä. (Dino 2012, 209). Toisaalta termiä käytetään yleisesti korostamaan tietyn suunnittelutavan sukupolvellista luonnetta.

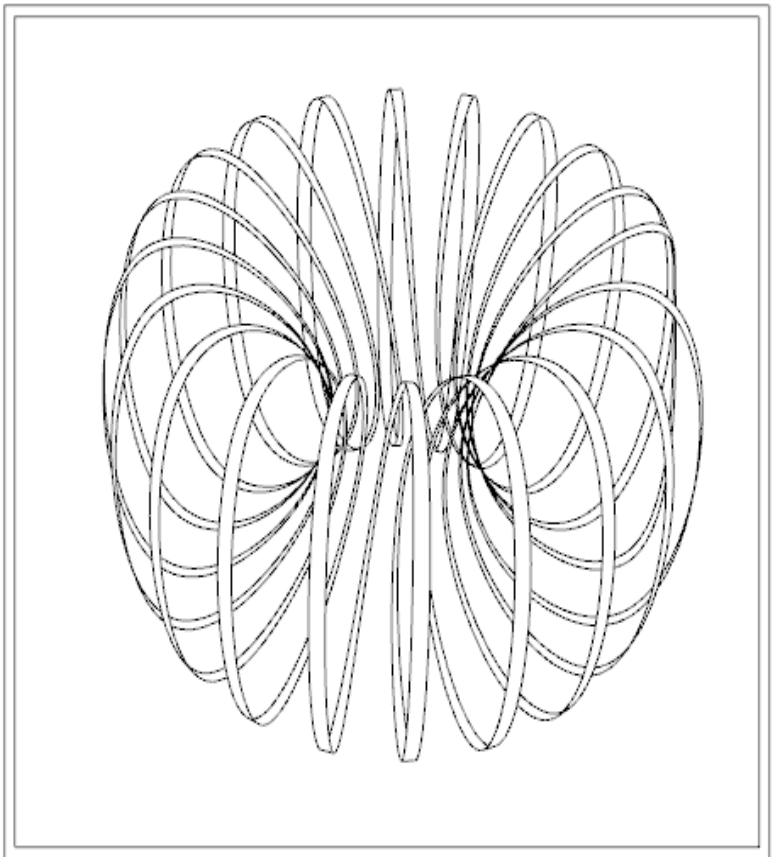
Grasshopper on 3D-mallinnusohjelma Rhinoceros 3D:n lisäosa ja visuaalinen ohjelmointikieli, jonka avulla tehdään algoritmista mallinnusta ja algoritmista suunnittelua. Grasshopperin on luonut David Rutten ja Robert McNeel & Associates.

Implisiittinen mallinnuksen avulla mallinnetaan riippuvuussuhteita ja objektin varsinaiset ominaisuudet määrittyvät suhteessa toisiinsa. Voidaan esimerkiksi määritellä malli, jossa ruuvin paikka ei ole eksplisiittisesti määritelty, vaan se seuraa kahvan mukana tätä siirrettäessä.

Rhinoceros (myös Rhino3D) on halpa ja käytetty pintamallinnusohjelma.

Parametri on määre, jonka avulla voidaan määritellä geometria, ilmiö tai järjestelmä (Tanska ja Österlund 2014, 13). Esimerkiksi ympyrän parametreja voisi olla halkaisija ja keskipisteen sijainti.

Parametrinen malli on termi, jota käytetään kuvaamaan mallia, joka on määritelty siten, että kokonaisuutta voi säätää parametrien arvoja muuttamalla. (Tanska ja Österlund 2014, 13) Parametrinen mallinnus on implisiittistä.



Kuva 1. Kuvan kierredonitsi olisi hankala mallintaa muuten kuin algoritmisesti. Vaikka soveltuvilla toisto-komennoilla muodon voisi mallintaa ei-algoritmisesti, olisi tehdyn muodon muokkaaminen hankalampaa ja vähemmän hallittua kuin algortimisesti mallinnetun kappaleen.

3 Algoritminen suunnittelu

Algoritmisesta suunnittelusta käytetään ristiin ainakin viittä termiä, joilla hieman eri painotuksella kuvataan samantyyppistä suunnittelua. Tässä luvussa perusteellaan miksi on valittu käyttää termiä algoritminen muotoilu kattokäsitteenä ilmiölle.

Samassa tai liki samassa merkityksessä keskenään käytetään sekaisin termejä parametrinen muotoilu *(Parametric design)*, generatiivinen muotoilu *(Generative design)*, laskennallinen muotoilu *(Computational design)*, algoritmiavusteinen muotoilu *(Algorithms-Aided Design)* ja algoritminen muotoilu *(Algorithmic design)*. Termeistä yleisimmin käytetty parametrinen muotoilu korostaa muodon parametrisointia, ja sitä, että parametrit ovat jälkeenpäin muutettavissa. Generatiivinen muotoilu korostaa muodon syntymistä, ja toisinaan evolutiivisten algoritmien käyttöä muotoiluongelman ratkonnassa. Algoritmiavusteinen muotoilu ja algoritminen muotoilu korostavat sitä, että eksplisiittisen mallin sijaan suunnitellaan kappaleen tuottava algoritmi. Tämän jälkeen suunnitteluongelma ratkotaan algoritmista mallia hyväksi käyttäen.

3.1 Algoritminen suunnittelu ja algoritminen mallinnus

Tässä kandidaatintyössä käytetään termiä algoritminen suunnittelu kattokäsitteenä kaikelle implisiittiselle muotoilulle. Kattokäsitteenä voisi käyttää termiä parametrinen muotoilu, mutta termin saaman kritiikin (mm. Tedeschi ym. 2014; Tanska ja Österlund, 2014) takia tässä työssä pitäydytään Arturo Tedeschin ym. (2014) lanseeraamassa termissä. Algoritminen muotoilu pitää sisällään generatiivisen muotoilun ja algoritmisesti toteutettavat parametriset mallit. Termi algoritminen muotoilu korostaa paradigman prosessillista luonnetta. Sen sijaan että keskitytään objektin attribuutteihin, pohditaan minkälainen prosessi tuottaisi artefaktin, joka toteuttaa artefaktille asetetut vaatimukset. Sen sijaan että luodaan ateria, tehdään resepti. Kun resepti on olemassa, voidaan sen parametreja muuttaa ja saada aikaan erilaisia aterioita.

Jotta voidaan algoritmin avulla ratkoa muotoiluongelmaa pitää se ensin parametrisoida, eli pitää määrittää mitkä muuttujat vaikuttavat kyseiseen muotoiluongelmaan. Kun muoto on parametrisoitu, luodaan algoritmi, joka mallintaa kappaleen parametrien mukaan, optimoi, varioi tai laskee muotoja tai arvoja. Lisäksi täytyy määrittää tavoitteet, joiden mukaan aikaan saatuja ratkaisuvaihtoehtoja arvioidaan. Tavoitteet voivat olla esimerkiksi eksplisiittisesti määriteltynä hyvyysfunktioksi, kuten evolutiivisia algoritmeja käytettäessä täytyy, tai olla silmämääräistä estetiikan arviointia.

Algoritminen mallinnus *(Algorithmic Modelling)* on tapa tuottaa 3d-malli algoritmisia välineitä käyttäen, eikä välttämättä vaadi algoritmista suunnittelua. Viiden, pinta-alaltaan vakion 17-21 -sakaraisten tähden suunnittelu käsin on helppoa. Toisaalta niiden mallinnus on helpompaa algoritmisesti kuin suoraan mallintamalla. Varsinkin, jos halutaan, että tähtien sakaroiden kulmat ovat

30 astetta, tai tiedetään, että sakaroita halutaan jälkikäteen muuttaa, helpottaa parametrinen malli muodon tuottamisessa.

3.2 Parametrinen mallinnus ja parametrinen suunnittelu

Muotoilijat käyttävä termiä parametrinen mallinnus *(Parametric Design)* kuvaamaan mallinnusta, jossa voidaan määrittää mallin osien geometriaa tai suhteita toisista osista riippuvaisina, ja jälkeenpäin muutettavina. Esimerkiksi käytettäessä solidimallinnusohjelmia CREO tai SolidWorks puhutaan usein parametrisesta mallinnuksesta (Shih 2013). Tässä yhteydessä parametrisuus kuvaa mahdollisuutta muuttaa parametreja ja siten mallin ominaisuuksia jälkikäteen. Esimerkiksi mallin komponentteja voidaan vaihtaa tai kappaleen seinämänvahvuutta voidaan muuttaa helposti ja nopeasti. Ei-parametrisessa mallinnuksessa muutokset saattavat vaatia sitä, että koko kappale joudutaan mallintamaan uudestaan.

Arkkitehdit käyttävät termiä parametrinen kuvaamaan tyylisuuntaa, jonka on väitetty alkaneen modernismin jälkeen. Tässä keskeistä ovat parametriset kokonaisuudet ja kokonaisuuksien suunnittelu. Zaha Hadidin suunnittelupartneri Patrick Schumacher toteaa parametrismin olevan uusi arkkitehtuurin tyylisuunta modernismin jälkeen. (Schumacher 2009). Kuitenkin parametrismi tyylisuuntana on saanut kritiikkiä osakseen. Jotkin kirjoittajat korostavat parametristen välineiden työkaluluonnetta ja varovat antamasta tyylisuunnan leimaa paradigmalle (Dino 2012, 216-217).

Termin parametrinen käyttö aiheuttaa haasteita, sillä sitä käytettään kuvaamaan useaa keskenään ristiriitaista ilmiötä. Siksi tässä kandidaatintyössä käytetään termiä algoritminen muotoilu kattokäsitteenä. Muun muassa matematiikassa määritellään parametri tarkoittamaan geometrian tai järjestelmän ominaisuuksia, joiden avulla tämä voidaan määritellä yksikäsitteisesti. Esimerkiksi kaksiulotteisessa avaruudessa olevan pisteen voi parametrisoida kahden muuttujan avulla. Matemaatiikassa käytetyn termin parametri määritelmän mukaan kaikki tietokoneella tehtävä suunnittelu on parametrista. Mukaan lukien sellainen eksplisiittinen suunnittelu, esimerkiksi vapaalla kädellä Photoshopissa piirtäminen, jota eivät arkkitehdit eivätkä muotoilijat kutsuisi parametriseksi.

3.3 Generatiivinen suunnittelu sekä Evolutiiviset ja geneettiset menetelmät

Termiä generatiivinen suunnittelu *(Generative design)* käytetään kuvaamaan algoritmista suunnittelutapaa, jossa korostetaan ratkonnan iteratiivista luonnetta. Generatiivisen mallin avulla tuotetaan luonnon genetiikkaa mukaillen sukupolvia, jotka aiempia paremmin ratkaisevat määritettyä suunnitteluongelmaa. Tällöin mallinnuksessa on keskeistä takaisinkytkentä: mallista n lasketaan data, joka ohjataan takaisin määrittämään uutta mallia n + 1.

Toisaalta termiä generatiivinen suunnittelu *(generative design systems, generative design)* käytetään kuvaamaan algoritmia tai toimintatapaa, joka tuottaa, generoi, artefaktin parametrien perusteella (Dino 2012, 209). Termiä käytetään siis sekä generoivassa, että sukupolvia *(generation)* korostavassa merkityksessä. Termin monimerkityksellisyyden takia generatiivinen suunnittelu lienee huono katokäsite.

Geneettiset ja evolutiiviset algoritmit ovat optimointimenetelmiä joiden avulla voidaan ratkoa moninaisia ja monimutkaisia optimointiongelmia (Tanska ja Österlund 2014, 12). Niitä voidaan käyttää osana algoritmista suunnittelua haluttaessa etsiä parempia ratkaisuja parametrisoituun suunnitteluongelmaan. Ongelmaan täytyy voida määrittää yksikäsitteinen hyvyysfunktio, funktio, jonka perusteella saadaan numeerinen arvio ratkaisun hyvyydestä. Hyvyysfunktion avulla määritellään optimoidaanko estetiikkaa, tuotantohintaa, lujuutta vai jotain muuta parametria tai parametriyhdistelmää. Esimerkiksi luvun 4.2 akustiikan optimointi lienee tehty geneettisiä algoritmeja hyväksi käyttäen.

Evolutiiviset ja geneettiset algoritmit ovat generatiivisen suunnittelun työkalu, ja siksi niistä ei ole kattokäsitteeksi. Samoin kuin generatiivista suunnittelua, määrittää evolutiivisia- ja geneettisiä algoritmeja takaisinkytkentä,. Evolutiivisten ja geneettisten algoritmien käyttöä voidaan pitää generatiivisen suunnittelun alakategoriana.

3.4 Laskennallinen suunnittelu

Termi laskennallinen suunnittelu *(Computational Design)* korostaa suunnittelutavan tietokoneavusteisuutta. Termi korostaa simuloinnin, robotiikan ja tekoälyn tarjoamia mahdollisuuksia suunnitteluongelmien ratkonnassa. Kouluja, joissa on laskennallisen suunnittelun koulutusohjelma on muiden muassa CMU School of Architecture ja UNSW Sydney (CMU School of Architecture 2017; UNSW Sydney 2017). Vaikka kaikki tietokonemallit ovat matemaattisesti määritettyjä, ja siten määritelmän mukaan laskennallista, ei laskennallinen suunnittelu soveltune kattokäsitteeksi. Esimerkiksi generatiivista suunnittelua ei määritä laskennallisuus, vaan ennemmin algoritminen lähestymistapa.

3.5 Yhteenveto käytetyistä termeistä

Termeistä parametrinen suunnittelu on saanut kritiikkiä osakseen, sillä on vaikea määritellä parametreja siten, ettei kaikki muotoilu olisi parametrista (mm. Tanska ja Österlund 2014). Generatiivinen muotoilu kuvaa vain osaa kentällä tehtävästä työstä, sillä kaikki generatiivinen muotoilu on parametrista ja algoritmista, mutta ei toisinpäin. Samoin perustein pitäydytään käyttämästä termiä laskennallinen suunnittelu. Tässä kandidaatintyössä käytetään siis termiä algoritminen muotoilu kattokäsitteenä kaikille yllämainituille suunnittelun muodoille.

4 Esimerkkejä ja etuja

Luvussa käsitellään esimerkkien avulla algoritmisen suunnittelun avulla saavutettavia hyötyjä. Laajan kuvan välittämiseksi on esimerkkejä koottu eri aloilta: arkkitehtuurista, pelisuunnittelusta, teollisesta muotoilusta ja korusuunnittelusta.

4.1 Algoritmisen suunnittelun edut

Algoritmista mallinnusta on käytetty tuottamaan estetiikaltaan omaleimaista arkkitehtuuria ja muotoilua. Algoritmista suunnittelua ei kuitenkaan pidä nähdä välineenä ainoastaan estetiikan luomiseen (Leach 2014). Algoritmista suunnittelua voidaan käyttää muotoilun tukena luotaessa luonnoksia tai optimoitaessa muotoa jonkun parametrien suhteen.

Algoritminen muotoilu on kattokäsite muotoilun paradigmalle, jossa muotoiluongelma ratkotaan määrittämällä algoritmi, jonka avulla voidaan löytää suunnitteluongelmaan ratkaisu. Sen sijaan että suunnitellaan esine, luodaan järjestelmä, joka tuottaa esineen annettujen lähtöarvojen perusteella. Lähtöarvoja muuttamalla voidaan tuottaa satoja tai tuhansia variantteja kappaleesta. Aikaan saaduista malleista voidaan laskea lujuusarvoja tai muuta hyödyllistä dataa, jota voidaan käyttää uusien mallien generoimiseen tai mallien arvioimiseen.

Algoritmisen suunnittelun avulla voidaan optimoida suorituskykyä (”Pioneering bionic 3D printing” 2017). Lisäksi evolutiivisten algoritmien avulla voidaan optimoida useita muotoiluparametrejä samanaikaisesti tai valjastaa tekoäly ratkomaan muotoiluongelmaa (Perez 2016). Toisaalta algoritmista suunnitelua voidaan käyttää luomaan omaleimaista estetiikkaa (”Maha Nakhon” 2017) tai lukemattomia muotovariaatioita (Nervous System inc. 2017).

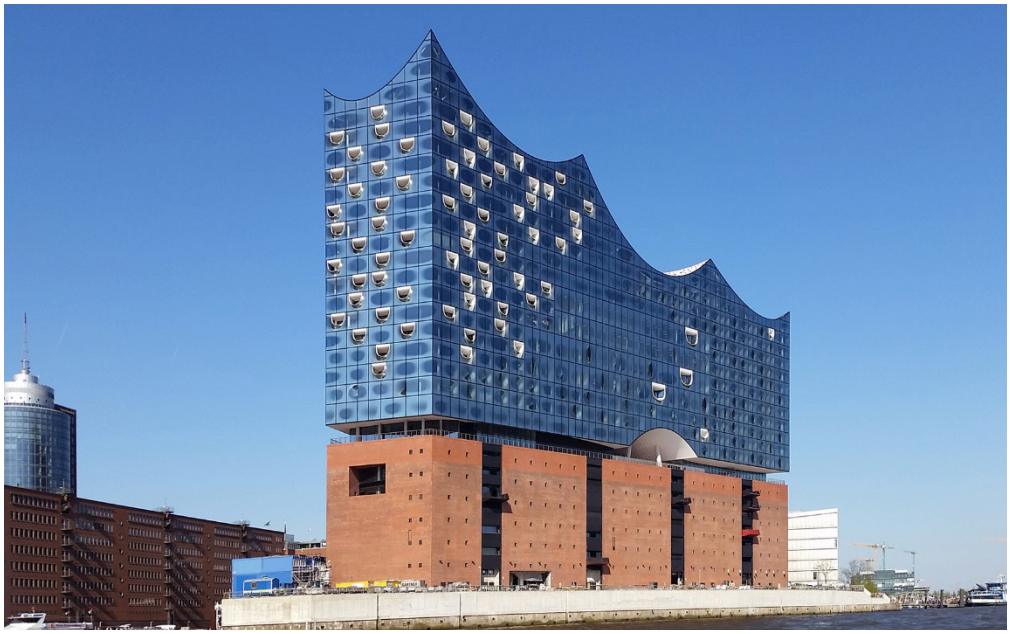
4.2 Algoritminen suunnittelu arkkitehtuurissa ja optimointi

Arkkitehdit ovat suunnitelleet algoritmisesti jo 20 vuotta, ja suunnittelutapa onkin viimeaikaisen teknisen kehityksen myötä tullut lähestyttävämmäksi ja yleistynyt (Frazer 2016). Arkkitehdit käyttävät tavasta termiä parametrinen suunnittelu.

Zaha Hadidin arkkitehtuuripartneri Patrick Schumacher kirjoitti 2008 Parametrisistimanifestin, jossa hän kuvaa parametrisismin olevan uusi arkkitehtuurin paradigma, jota tulisi noudattaa kaikilla arkkitehtuurin osa-alueilla. Schumacher korostaa, että systemaattinen, jatkuva variointi ja dynaaminen, parametrinen muotoilu koskettaa kaikkea muotoilua arkkitehtuurista esinemuotoiluun. Hän korostaa, ettei variaatiota pidä tuottaa variaatioiden itsensä tähden, vaan hallitusti dynaamisen muutoksen avulla löytää parhaita ratkaisuja. Tavoitteena algoritmisen muotoilun avulla voidaan saavuttaa kompleksisuutta, joka silti on ymmärrettävää. (Schumacher 2009)

Schumacher toteaa 20 vuoden tutkimuksen jälkeen parametrisismi on kypsä vastaamaan sille asetettuihin haasteisiin. Se mahdollistaa älykästä työnjakoa ja yhteiskuntaa leimaavan laskennallisuuden hyödyksi käyttämistä rakenteita suunniteltaessa. Schumacher ehdottaa parametrisismin olevan modernismin ja postmodernismin jälkeen alkava uusi arkkitehtuurin tyylisuunta. (Schumacher 2016). Parametrisimista tyylisuuntana kuitenkin keskustellaan, eikä selvään lopputulokseen olla päädytty (Leach 2014).

Hyvä esimerkki algoritmien käytöstä suunnittelun apuna on konserttitalo Elbphilharmonie Hampurissa (kuva 2). Sen akustiikka on toteututettu algoritmeja avuksi käyttäen. Konserttitalon 2100 paikkaisen pääsalin sisäseinät koostuvat 10 000 yksilöllisestä CNC-koneistetusta paneelista, joissa jokaisessa on miljoona uurretta. Uurteet ovat paneelin sijainnista riippuen 4-16 cm syviä, ja hajottavat paneeliin osuvat ääniaallot. Paneelit on suunnitellut Yasuhisa Toyota, jonka käsialaa on myös Helsingin Musiikkitalon akustiikka (”Konserttisali – musiikin viinitarha” 2017). Elbphilharmonien akustiikkaa ei olisi voitu optimoida näin muuten kuin algoritmisesti. (Stinson 2017).



Kuva 2. 2016 valmistunut konserttitalo Elbphilharmonie Hampurissa on suunniteltu algoritmeja avuksi käyttäen. (Hamburg Elbphilharmonie 2017)

Ole Scheeren suunnittelema 2016 valmistunut Maha Nakhon (kuva 3, sivu 12) on Thaimaan korkein rakennus (MahaNakhon 2017). Sen ikään kuin pikselöitynyt ulkomuoto olisi eksplisiittisesti mallinnettaessa työläs tehdä ja vielä työläämpi varioida, mutta algoritmisesti muotoiltaessa rakennuksen muoto saadaan helposti ns. attraktoreilla aikaan, määrittämällä käyrä kuutioista koostuvan tornin ympärille ja poistamalla kuutioita käyrän läheisyydestä. Algoritminen muotoilu mahdollistaa eksplisiittiseen mallinnukseen verrattuna uusia tapoja automatisoida toimintoja ja luoda variaatiota.

Kuva 3. Bangkokissa sijaitseva Maha Nakhon on helposti mallinnetavissa algoritmisesti. Eksplisiittisesti mallinnettaessa varsinkin variointi olisi vaikeaa. (Maha Nakhon, 2017)

4.3 No Man´s Sky ja variaatio

Hello Gamesin pelissä No Man´s Sky (Hello Games 2017) on algoritmisen mallinnuksen avulla luotu 18 triljoonaa planeettaa. Algoritmi generoi planeetat ja niille kasvi- ja eläinlajeja. Pelissä seikkaillaan avaruudessa ja tutkitaan planeettoja joilla kukaan muu pelaaja ei ole käynyt. Jokainen kasvilaji, olentotyyppi ja geologinen muodostelma on uniikki. (”No Man’s Sky” 2017).

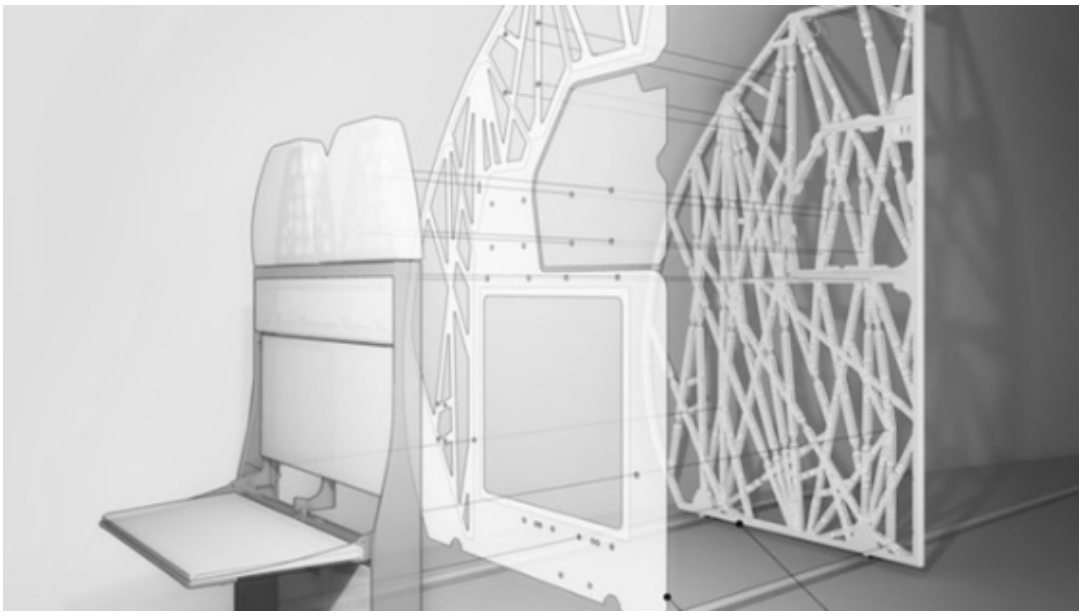
No Man´s Sky on äärimmäinen esimerkki algoritmisesti saavutettavasta variaatioiden määrästä. Jos sisältöä ei luotaisi algoritmisesti tarvitsisi peliä varten massiivisen tietokannan, jossa olisi jokaiselle planeetalle, ja jokaiselle eliölle jokaisella planeetalla 3d-mallit, teksturointi, animaatiot ja ominaisuudet interaktioihin pelaajan kanssa. No Man´s Sky tiedoston koko on kuitenkin vain 6 GB. Vertailun vuoksi Fifa 2017 vaatii minimissään 44 GB tallennustilaa (”Fifa 2017” 2017). (”No Man’s Sky” 2017).

Kuva 4. No Man´s Sky Logo. (“No Man’s Sky” 2017)

4.4 Tekoälyavusteinen suunnittelu Autodesk Dreamcatcherin avulla

Autodeskin valmistama Dreamcatcher (”Project Dreamcatcher” 2017) on neuroverkkoja hyödyntävä muotoilutyökalu, jolle syötetään muotoiltavalle kappaleelle asetettavia vaatimuksia ja/tai rajoitteita. Näitä voivat olla esimerkiksi lujuus, paino, aerodynamiikka, valmistusmateriaalit, valmistusmenetelmät tai ennalta määrätyt liitokset. Näiden pohjalta ohjelma laskee tuhansia vaihtoehtoja ja oppii tuottamaan yhä parempia malleja muotoiluongelman ratkaisuksi. Evolutiivisten algoritmien avulla voidaan huomioida useita keskenään ristiriitaisiakin tavoitteita samanaikaisesti. Ohjelman annettua useita vaihtoehtoja muotoilija voi lopulta verrata malleja ja näiden ominaisuuksia ja valita parhaaksi katsomansa. Koska evolutiiviset algoritmit noudattavat samoja periaatteita kuin luonnonvalinta voi evolutiivisten algoritmien aikaansaama lopullinen muoto voi olla ulkonäöltään kovin orgaaninen.

Airbus A320 konseptilentokoneessa on Dreamcatcherin avulla optimoituja tilanjakajia (kuva 5). Uudet ovat aiempia lujempia vaikka painavat 45 % vähemmän kuin aiemmat. Lisäävien valmistusmenetelmien avulla materiaalia kuluu 95 % vähemmän. Osan käyttö säästää arviolta 3 180 kg polttoainetta vuodessa, ja lentokoneessa tilanjakajia on neljä. (”Pioneering bionic 3D printing” 2017).



Kuva 5. Airbus A320 osa painaa 45% vähemmän kuin aiempi ja säästää siten polttoainetta. (”Pioneering bionic 3D printing” 2017)

Kuva 6. Dreamcatcherin avulla painon ja lujuuden suhteen optimoituja kappaleita. Vasemmalla on alkuperäinen komponentti, joka painaa 300% enemmän kuin optimoitu. (Perez 2016)

Dreamcatcherin tapaisten evolutiivisten algoritmien avulla voidaan löytää muotoiluratkaisuja, joita ei muuten oltaisi voitu löytää. Niitä voidaan käyttää yleiseen optimointiin tai muotojen variointiin, ja ohessa syntyy estetiikaltaan orgaanisia muotoja. Kuvassa 6 (sivu 14) on liitoskomponentti, joka on optimoitu rasituksen ja painon suhteen. Materiaalia on käytetty ainoastaan siellä missä sitä tarvitsee käyttää. Kuvassa vasemmalla on alkuperäinen, jota evolutiivisen algoritmin avulla optimoitiin. Algoritmin avulla aikaan saatu painaa neljänneksen alkuperäisen painosta vaikka on yhtä vahva. (Perez 2016).

4.5 Massakustomointia ja uniikkeja esineitä

Design Morphine (Designmorphine 2017) ja Nervous System (Nervous System inc. 2017) molemmat myyvät uniikkeja koruja. He ovat muotoilleet yksittäisen esineen sijaan algoritmin, joka tuottaa tyyliltään samanlaisia, mutta yksilöllisiä esineitä. Esineet tuotetaan lisäävien valmistustekniikoiden avulla tai fotoetsaamalla, käyttäen tulostusta hyväksi. Kuvassa 7 on Nervous Systemin tuottama kaulakoru.

Kuva 7. Nervous Systemin korut ovat uniikkeja. Kuvassa coral series #1060. (Nervous System inc. 2017)

5 Grasshopper ja sen käyttö

Kandidaatintyön aikana tuotin muotoilijoille suunnatun opiskelumateriaalin Grasshopperin opettelua varten. Materiaalia käytettiin opetuksen tukena algoritmista muotoilua käsittelevällä kurssilla keväällä 2017 Aalto-yliopiston muotoilun laitoksella. Tässä luvussa avataan Grasshopper -ohjelman käyttöä ja luotua opetusmateriaalia.

Opetusmateriaalin tavoitteena on toimia opiskelun ja opetuksen tukena antamalla kirjallinen tukimateriaali kertaukseksi, ja toisaalta mahdollistaa itseopiskelu materiaalin pohjalta. Yhdessä opetusmateriaalin opiskelu ja tähän kandidaatintutkielmaan tutustuminen antaa varsin hyvän kuvan algoritmisesta suunnittelusta, sen eduista ja käytännön toteutuksesta.

5.1 Grasshopper

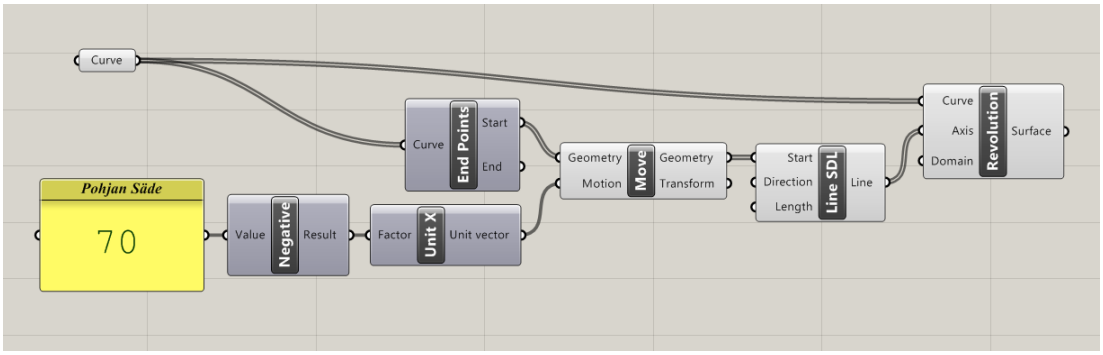
Grasshopper on ilmainen lisäosa suosittuun McNeelin Rhinoceros pintamallinnusohjelmaan. Grasshopperissa ohjelmoidaan visuaalista ohjelmointikieltä, jossa tekstin sijaan operoidaan visuaalisilla funktioilla.

Grasshopperin käyttö ei vaadi aiempaa ohjelmointiosaamista. Grasshopperin avulla saa helposti visualisoitua tuottamiaan malleja ja siksi sen avulla on helpompi lähestyä muotoiluongelmia kuin olisi perinteisten ohjelmointikielten avulla ohjelmoitaessa. (Mode Lab 2015, 4-8)

Grasshopper on ilmaisuudestaan huolimatta huomattavan vahva ja monipuolinen työkalu. Siihen on helppo luoda lisäosia, ja niitä onkin eri tarkoituksia varten toista sataa. Lisäosista esimerkiksi Kangaroo mahdollistaa fysiikkamallinnusta, Elkin avulla voidaan tuoda Open Street Map -dataa mallinnusohjelmaan. Octopus mahdollistaa evolutiiviset algoritmit ja Dragonfly animaatioiden teon. Jos jotakin toimintoa ei voi suoraan ohjelmoida Grasshopperissa voi apuna käyttää C# tai Python -ohjelmointikieliä. (Food4rhino 2017).

5.2 Grasshopper käytännössä

Kuten todettua operoidaan Grasshopperissa visuaalista käyttöliittymää johon ohjelmoidaan visuaalista koodia. Kuvassa 8 (sivu 17) on esitettynä hyvin yksinkertainen pyörähdyskappaleen tuottava Grasshopperkoodi. Vasemmalla ylhäällä olevaan komponenttiin *(Curve)* tuodaan Rhinosta pyörähdyskappaleelle profiili. Vasemmalla, keltaiseen laatikkoon sijoitetaan numero, tässä 70, jonka avulla määritetään kappaleen pohjan säde. Koodi tuottaa syötettyjen tietojen avulla maljakon.



Kuva 8. Esimerkki lyhyestä Grasshopperkoodista. Koodi tuottaa pyörähdyskappaleen.

Oikeanpuoleisin komponentti *(Revolution)* tekee varsinaisen pyöräytyksen. Viisi keskellä olevaa komponenttia määrittävät minkä pisteen läpi kulkee akseli, jonka ympäri pyörähdyskappale pyöräytetään.

Esimerkin Grasshopperkoodissa on 8 komponenttia. Liitteessä 1 olevan ohjeen mukaisessa kierredonitsikoodissa niitä on toista sataa.

5.3 Listat Grasshopperissa

Avain Grasshopperin tehokkaaseen käyttöön on Grasshopperin tietorakenteiden ymmärtäminen. Grasshopperissa tieto on puurakenteissa (Data Tree). Puurakenteet ovat listoja, joissa listan alkioina voi olla listoja. Listan avulla voidaan yhtä komponenttia käyttämällä laskea arvoja kaikille listan luvuille. Voidaan esimerkiksi tuoda edellisen luvun pyörähdyskappalegeneraattoriin pohjan säteeksi yhden luvun sijaan lista, jossa on arvot 45, 70, 102 ja 140 ja luoda vastaavan säteiset maljakot. Jos listassa olisi neljän luvun sijaan sata lukua tekisi ohjelma sata maljakkoa.

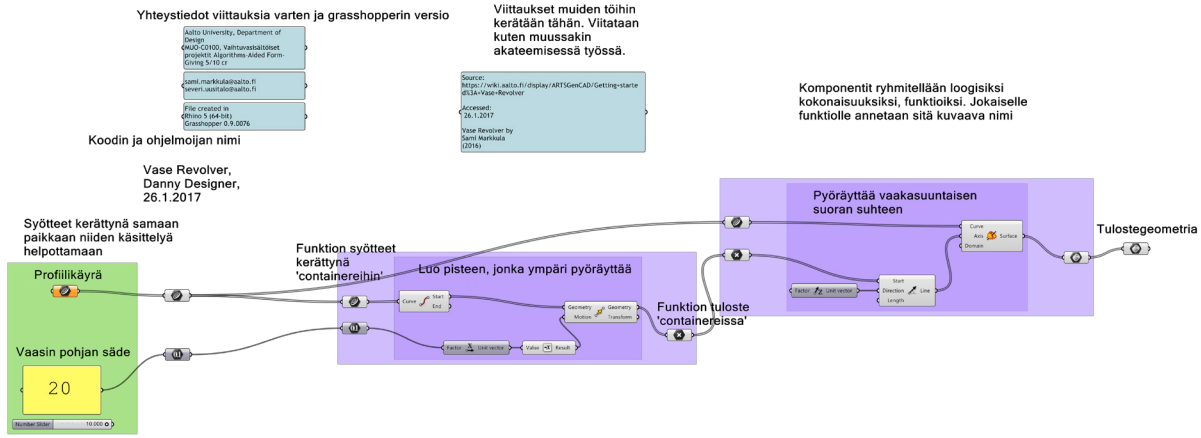
Algoritmisen lähestymistavan seurauksena voidaan variaatioita helposti lisätä muuttamalla syötetteen parametreja. Monimuotoisuuden lisääminen on helppoa muuttamatta ohjelmaa. Jos on esimerkiksi luotu kattava malli maljakosta, voidaan sen ominaisuuksia hienosäätää parametrejä muuttamalla. Toisaalta kaikki tieto on laskennallisessa muodossa, joten samalla voidaan laskea lujuusarvoja kaikille sadalle maljakolle ja järjestää ne paremmuusjärjestykseen esimerkiksi lujuuden tai materiaalinkulutuksen suhteen.

5.4 Kommentointiohje ja hyvä kommentointitapa

Kandidaatintyön aikana määritin Delftin teknillisen yliopiston hyvän Grasshopper-ohjelmointitavan ohjeiden (”Grasshopper ”Good Practice” 2017) sekä Danny Boyesin ohjelmointitavan (”Organizing GH documents” 2017) pohjalta Grasshopper-ohjelmointiohje Aalto-yliopiston Muotoilun laitokselle. Noudattamalla ohjetta suunnittelijan tuottama koodi on paremmin jäsenneltyä ja helpommin luettavaa kuin vapaasti tuotettu koodi. Yhtenäinen ohjelmointitapa laitoksen

opiskelijoiden välillä helpottaa muiden kirjoittaman koodin lukemista ja muiden koodeista oppimista. Myös ryhmässä työskentely helpottuu yhtenäisen koodaustavan myötä.

Erona Delftin teknillisen yliopiston ohjelmointitapaan (”Grasshopper ”Good Practice”” 2017), kaikki komponentit ryhmitellään yhteen toiminnallisiksi funktioiksi, joilla kaikilla on omat syötteensä ja tuloste. Tällöin, sen lisäksi, että komponenttien toiminta on helpommin luettavissa, voidaan ohjelmasta helposti kopioida kokonaisia komponenttiryhmiä muualle käytettäväksi. Lisäksi, muotoilun laitoksella opettamamme tapa on joustavampi, varsinkin monimutkaisia ohjelmia tehtäessä, sillä funktioiden välisten yhteyksien muuttaminen on helpompaa.

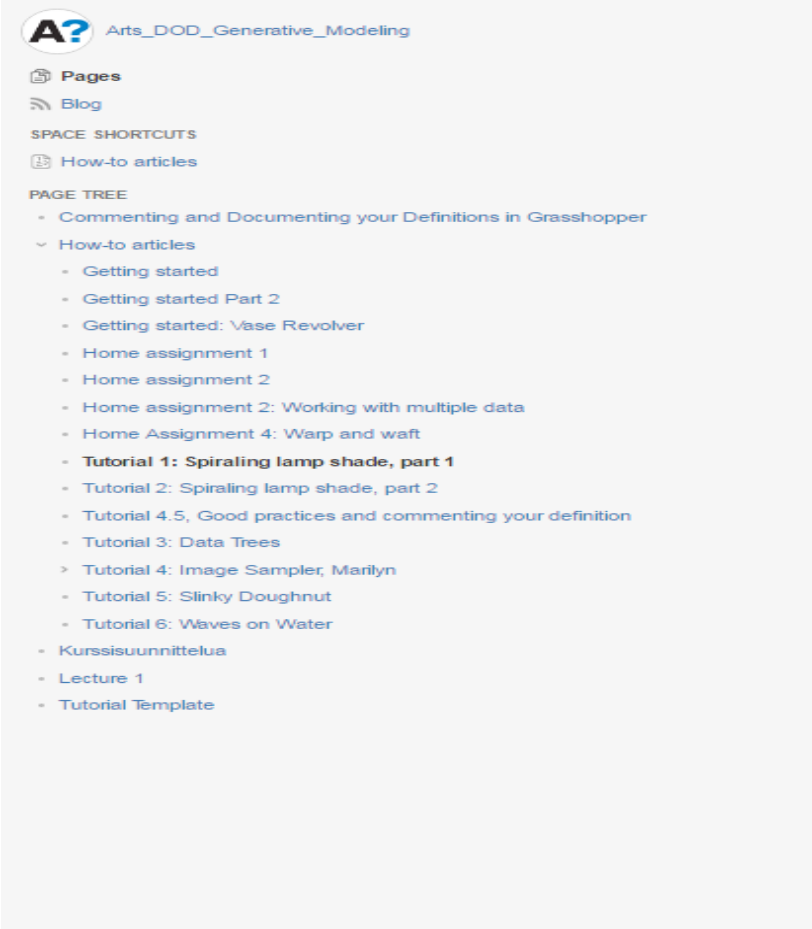


Kuva 9. Ote opiskelumateriaalista. Kommentointiohje.

Kuvassa 9 on kommentoituna aiemmin esitelty pyörähdyskappaleen tuottava koodi (kuva 8, sivu 17). Kuvassa 9 nähdään keskellä ja oikealla kaksi erillistä, useasta komponentista koostuvaa funktiota rajattuna liloihin laatikoihin. Lilojen laatikoiden sisään jäävät komponentit muodostavat ikään kuin uudet funktiot. Saatuja funktioita voidaan käsitellä uusina komponentteina, joilla on omat syötteensä ja tulosteensa. Ohjelmointitavassa kaikki muutettavat parametrit on kerätty vasempaan reunaan, jotta niitä ei tarvitse etsiä koodin joukosta.

5.5 Algorithms-Aided Form-Giving -kurssin opetusmateriaali

Kandidaatintyönä suunnittelin opetusmateriaalin Aalto-yliopiston kurssille *Algorithms-Aided Form-Givin*g (algoritmiavusteinen muodonanto, AAFG). Opetusmateriaalia varten tutustuin arkkitehtuurin laitoksen kurssiin *Design of Structures Independent Study: Parametric Architecture* (Wikar ja Österlund 2016), jolla käsiteltiin Grasshopperin käyttöä. Kurssin tehtävät oli ymmärrettävästi kohdennettu soveltamaan algoritmeja arkkitehtonisten kohteiden suunnitteluun.



Suunniteltaessa AAFG-kurssia haluttiin tehtävien antavan työkaluja muotoilutyöhön. Tehtäviä piti luoda tai soveltaa sopimaan muotoilun mittakaavaan.

Opetusmateriaalia varten luotiin wiki-sivusto (Markkula 2017), johon laaditut ohjeet koottiin. Kurssin edetessä julkistettiin wikistä sivuja kotitehtäviksi ja kotitehtävien tueksi. Vieressä on kuvassa 10 esitettynä wikin sisällysluettelo. Kussakin ohjeessa on sivuja 6-12. Liitteenä 1 on esimerkkinä ohje *Tutorial 5: Slinky Doughnut.*

Kuva 10. Wiki-sivuston sisällysluettelo. Wiki sisältää kotitehtäviä, koodausohjeita sekä ohjeet Grasshopperin asentamiseksi ja käyttövalmiiksi saattamiseksi.

Vertailukohdiksi valitut ohjeet kertovat vaiheittain askel askeleelta oikeiden funktioiden lisäämisen koodiin halutun lopputuloksen saamiseksi (mm. Reilly 2014; ”The Grasshopper Primer” 2017; Tedeschi ym. 2016). Usein jätetään kertomatta miksi kukin funktio sijoitellaan paikalleen. Vertailukohtien mukaisten ohjeiden mukaan saa tuotettua halutun lopputuloksen, mutta usein taustalla piilevät periaatteet jäävät huomaamatta.

Kurssin AAFG ohjeet tein seuraamaan luontevaa ja johdonmukaista ongelmanratkaisua:

1. esitetään mitä tahdotaan saada aikaan
2. esitetään luonteva ratkaisumalli

3a. jos ilmenee ongelmia ratkotaan ongelmat

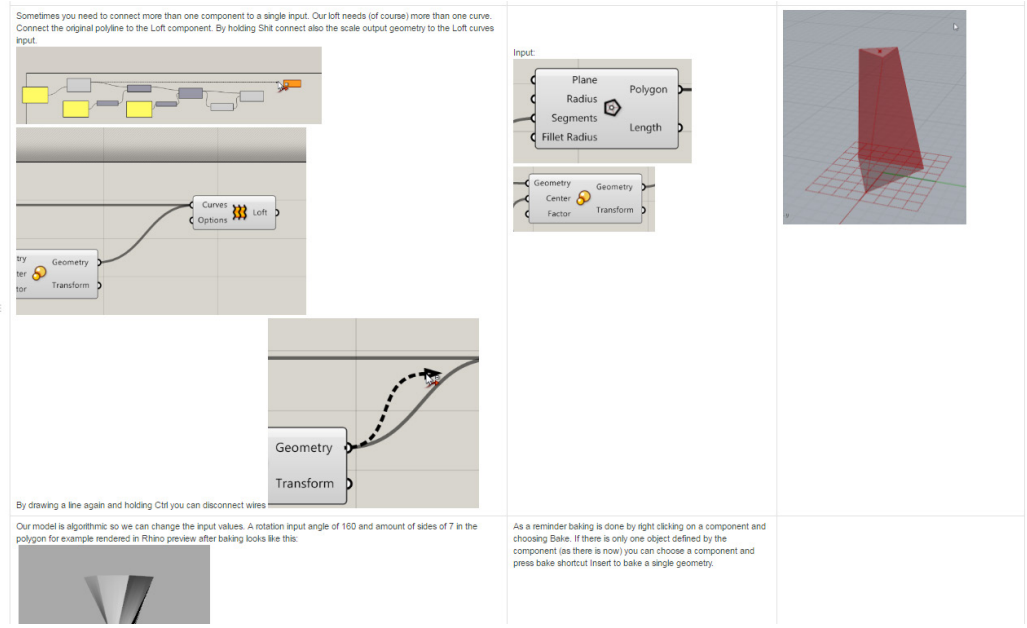
3b. jos ongelmia ei ilmene jatketaan eteenpäin.

Tyypillisessä ohjeessa esitetään suoraan toimivaa koodia eikä virheitä synny (Reilly 2014; ”The Grasshopper Primer” 2017; Tedeschi ym. 2016). Tällöin tietysti saa nopeammin halutun toimivan koodin aikaan, mutta samalla ei pääse ratkomaan virheitä joita syntyy koodia suunniteltaessa väistämättä. Koska kurssin opetustavoitteena oli mahdollistaa itsenäinen työskentely Grasshopperin avulla valittiin ohjeen toteutustapa opettamaan virhetilanteiden korjaamista samalla kuin se opettaa Grasshopper-ohjelmointia.

Taulukossa 1 on havainnkuva kolmisarakkeisen ohjeen rakenteesta. Vasempaan searakkeeseen sijoitettiin kirjallinen selostus siitä, mitä halutaan saada aikaan. Siinä on myös kuvallinen ohje vaiheen toteuttamiseksi, helpottamaan Grasshopperissa navigointia. Keskellä on lisäohjeita ja kommenteja. Oikealla on näkymä Rhinossa . Kuvassa 11 on esimerkkinä ote opetusmateriaalista. Kattavampi esimerkki löytyy liitteestä 1.

Taulukko 1. Taulukossa esitetään ohjeen rakenne sarakkeittain.

Vasen sarake	**Keskisarake**	**Oikea sarake**
Tavoite sekä ohjeet vaiheen toteuttamiseen	Lisätietoa ja havainnollistavia kuvia. Tarpeen mukaan linkkejä ja lähteitä lisäopiskeluun	3D-näkymä Rhinossa aikaansaadusta geometriasta



Kuva 11. Esimerkki ohjeesta. Oikealla on 3d-näkymä Rhinosta, vasemmalla kuvakaappauksia Grasshopperista. Keskisarakkeessa näkyy syötettä havainnollistavia kuvia ja lisäohjeita.

Noudattamalla ohjetta saadaan tuotettua algoritmisesti toteutettu muoto, esimerkiksi ohjeen 5 (liite 1.) avulla saadaan kuvan 1 (sivu 6) mukainen kierredonitsi aikaan. Ohjetta noudattaen oppii siis tuottamaan mallin mukaisen muodon, mutta ehkä tärkeämpänä ajattelemaan muotoilua algoritmisesti. Ohjetta käytettiin AAFG -kurssilla sekä itsenäisten kotitehtävien antoon, että soveltuvin osin runkona luennoilla toteutettaville harjoituksille.

6 Keskustelu

Algoritmisesti suunniteltaessa siirtyy fokus suunnitteluongelman ominaisuuksiin esineen ominaisuuksien sijaan. Sen sijaan, että iteroidaan ratkaisuvaihtoehtoja, iteroidaan menetelmää joka tuottaa ratkaisun. Ongelman reunaehtojen määrittely korostuu; on tärkeää määritellä mitä lopputulokselta haluaa.

Esimerkiksi ratkottaessa luvun 4.4 Airbusin väliseinän muotoiluongelmaa tärkeitä parametreja ovat paino sekä lujuus tilanjakajaan kohdistuvien rasitusten suhteen. Reunaehtoja oli tilanjakajan fyysiset mitat, tuotantomenetelmä ja liitosten sijainnit. Näiden tietojen pohjalta määritettiin evolutiivinen algoritmi, joka tuottaa väliseiniä. Väliseinän ominaisuudet, ainevahvuudet ja mitat seuraavat algoritmin määrittelystä ja ovat implisiittisiä. Ollaan ensisijaisesti määritelty suunnitteluongelmaa, jonka seurauksena syntyy ratkaisu.

Toisaalta simuloimalla tuotetun ratkaisuvaihtoehdon käyttöä voidaan havaita puutteita ongelmanmäärittelyssä ja reunaehtoja voidaan muuttaa havaintojen pohjalta. Varsinainen suunnittelu tapahtuu reunaehtojen ja tavoitteiden sekä algoritmin määrittelyn avulla. Näiden pohjalta algoritmi tuottaa ratkaisun, jonka hyvyys seuraa määritetyistä tavoitteista ja reunaehdoista.

Kuten aiemmin todettu, algoritmisesti suunniteltaessa täytyy tavoitteet ja reunaehdot tuotteelle määritellä eksplisiittisesti. Suunnitteluvaiheessa arvioitaessa ratkaisuvaihtoehtoa arvioidaan algoritmin tuotosta, joka toisaalta pohjaa suunnittelijan määrittämiin reunaehtoihin ja hyvyysfunktioon. Koska ratkaisun hyvyys riippuu reunaehtojen ja hyvyysfunktion määrittelyistä, saadaan parempi lopputulos paremmilla määrittelyillä.

Grasshopper on vahva työkalu optimointiin ja algoritmiseen mallintamiseen. Grasshopperin, ja yleisemminkin algoritmisen suunnitteluprosessin, haasteita on tarve kattavasti parametrisoida suunnitteluongelma tai ratkaisu siihen. Ehkä tulevaisuudessa osa parametrisoinnista voidaan älykkäiden ohjelmistojen avulla automatisoida.

Parametrisointi pakottaa ymmärtämään ratkaisua uudella tavalla. Vielä: parametrisointi voidaan nähdä suunnittelun formalisointina, sen työkaluna tai kielenä. Saman ongelman erilaisilla parametrisoinneilla voidaan saada aikaan toisistaan poikkeavat lopputulokset. Toisaalta formalisoinnin avulla on mahdollista löytää ratkaisuja joita ei olisi ilman formalisointia helppoa löytää.

Jotkut suunnittelukohteet varmastikin ovat toisia otollisempia ratkoa algoritmisesti. Kiinnostavaa on tutkia mistä suunnitteluongelman ominaisuuksista seuraa, että algoritmiset menetelmät ovat kustannustehokas työkalu. Ainakin suorituskykykeskeisissä toimeksiannoissa joissa optimointi on tärkeimpiä kriteereitä ratkaisun hyvyydelle, on evolutiivisten menetelmien käyttö perusteltua (Dino 2012, 220-221).

On aihetta tutkia missä vaiheissa suunnitteluprosessia algoritmiset suunnittelumenetelmät ovat perusteltuja ja miten integroida niitä olemassa olevaan suunnitteluprosessiin. Tietysti missä vaiheessa vaan algoritmista suunnitteluprosessia voidaan ottaa algoritmin ehdottama ratkaisu ja siirtää se osaksi perinteistä suunnitteluprosessia. Tällöin ratkaisuun voi suhtautua ikään kuin luonnoksena, jonka pohjalta voidaan jatkaa iterointia.

Algoritminen suunnitteluparadigma sinällään muuttanee tapaamme muotoilla (Dino 2012, 220). Nähtäväksi jää mitä mahdollisuuksia neuroverkkojen ja tekoälyn yhdistäminen helppokäyttöisiin algoritmisiin suunnitteluohjelmistoihin tarjoaa.

7 Yhteenveto

Algoritminen suunnittelu on paradigma, jota Arkkitehdit ovat harjoittaneet jo kaksikymmentä vuotta. On alettu puhua uudesta tyylisuunnasta, parametrisismista. Esimerkkejä algoritmien hyödyntämisestä löytyykin paljon juuri arkkitehtuurin parista.

Myös tuotemuotoiluun algoritminen muotoilu avaa monia uusia mahdollisuuksia. Voidaan tuottaa muotoja, jotka eivät olisi mahdollisia muuten. Algoritmisen suunnittelun avulla voidaan tuottaa laskennallisia malleja, joiden avulla voidaan optimoida haluttujen parametrien suhteen. Toisaalta algoritmisen suunnittelun avulla voidaan hyödyntää evolutiivista optimointia tai tekoälyä suunnittelun apuna.

Tämän kandidaatintyö antaa varsin hyvän kuvan algoritmisen suunnittelun nykytilasta. Tutustumalla kandidaatintutkielman aikana tuotettuun opetusmateriaaliin saa valmiudet tehdä algoritmista mallinnusta käytännössä. Tutkielmaa voidaankin sinällään käyttää osana algoritmisen suunnittelun opetusta.

Lähteet

CMU School of Architecture. (2017). *Computational Design*. Saatavissa: https://soa.cmu.edu/computational-design/ [Viitattu 15.4.2017].

Designmorphine. (2017). Saatavissa: http://designmorphine.org/ [Viitattu 11.4.2017].

Dino, G. (2012). Creative Design Exploration By Parametric Generative Systems In Architecture. *METU JOURNAL OF THE FACULTY OF ARCHITECTURE*.

En.wikipedia.org. (2017). *Generative Design*. Saatavissa: https://en.wikipedia.org/wiki/Generative_Design [Viitattu 19.4.2017].

Fifa 2017 (2017). Saatavissa: https://store.playstation.com/#!/en-us/games/ea-sports-fifa-17-standard-edition/cid=UP0006-CUSA03257_00-FIFAFOOTBALL2017 [Viitattu 15.3.2017].

Food4Rhino. (2017). *Food4Rhino*. Saatavissa: http://www.food4rhino.com/ [Viitattu 15.4.2017].

Frazer, J. (2016). Parametric Computation: History and Future. *Architectural Design*, 86(2), pp.18-23.

Grasshopper "Good Practice" - TOI-Pedia. (2017). Saatavissa: http://wiki.bk.tudelft.nl/toi-pedia/Grasshopper_%22Good_Practice%22 [Viitattu 15.3.2017].

Hamburg Elbphilharmonie. (2017). Saatavissa: https://upload.wikimedia.org/wikipedia/commons/0/0a/Hamburg_Elbphilharmonie_2016.jpg [Viitattu 15.3.2017].

Hello Games. (2017). Saatavissa: http://www.hellogames.org/) [Viitattu 15.3.2017].

Konserttisali – musiikin viinitarha. (2017). Saatavissa: https://www.musiikkitalo.fi/fi/artikkelit/konserttisali-musiikin-viinitarha [Viitattu 15.3.2017].

Leach, N. (2014). Parametrics Explained. *Next Generation Building*, 1(1).

Maha Nakhon. (2017). Saatavissa: https://en.wikipedia.org/wiki/MahaNakhon [Viitattu 15.3.2017].

Mode Lab. *The Grasshopper Primer*. (2015). Saatavissa: http://grasshopperprimer.com/en/index.html [Viitattu 5.4.2017].

No Man's Sky. (2017). *No Man's Sky Homepage*. Saatavissa: http://www.no-mans-sky.com/ [Viitattu 15.3.2017].
Organizing GH documents – Grasshopper. (2017). Saatavissa: http://www.Grasshopper3d.com/m/discussion?id=2985220%3ATopic%3A952293 [Viitattu 15.3.2017].

Perez E. (2016). *The Alien Style of Deep Learning Generative Design – Intuition Machine*. Saatavissa: https://medium.com/intuitionmachine/the-alien-look-of-deep-learning-generative-design-5c5f871f7d10#.bi2noukxe [Viitattu 15.3.2017].

Pioneering bionic 3D printing Learning from nature. (2017). Saatavissa: http://www.airbus-group.com/int/en/story-overview/Pioneering-bionic-3D-printing.html [Viitattu 15.3.2017].

Project Dreamcatcher. (2017). Saatavissa: https://autodeskresearch.com/projects/dreamcatcher [Viitattu 15.3.2017].

Reilly, C. (2014). *Learning Grasshopper*. www.Lynda.com. Saatavissa: https://www.lynda.com/Grasshopper-tutorials/Up-Running-Grasshopper/174491-2.html [Viitattu 16.4.2017].

Schumacher, P. (2009). Parametricism: A New Global Style for Architecture and Urban Design. *Architectural Design*, 79(4), s.14-23.

Schumacher, P. (2016). Parametricism 2.0: Gearing Up to Impact the Global Built Environment. *Architectural Design*, 86(2), s.8-17.

Shih, R. (2013). *Parametric modeling with Creo Parametric 2.0*. 1. painos. Mission, Kansas: SDC Publications.

Stinson, L. (2017). *What Happens When Algorithms Design a Concert Hall? The Stunning Elbphilharmonie*. WIRED. Saatavissa: https://www.wired.com/2017/01/happens-algorithms-design-concert-hall-stunning-elbphilharmonie [Viitattu 15.3.2017].

Tanska, T. ja Österlund, T. (2014). *Algoritmit puurakenteissa*. 1. painos. DigiWoodLab Oulun yliopisto, Arkkitehtuurin tiedekunta.

Tedeschi, A., Andreani, S. and Wirz, F. (2016). *AAD_Algorithms-Aided Design*. 1. painos. Brienza: Le Penseur Publisher.

UNSW Sydney. (2017). *Computational Design Program Details*. Saatavissa: https://www.be.unsw.edu.au/undergraduate-degrees/computational-design/program-details [Viitattu 15.4.2017].

Wikar ja Österlund. (2016) Luentosarja *Design of Structures Independent Study: Parametric Architecture*. Aalto-yliopisto, arkkitehtuurin laitos

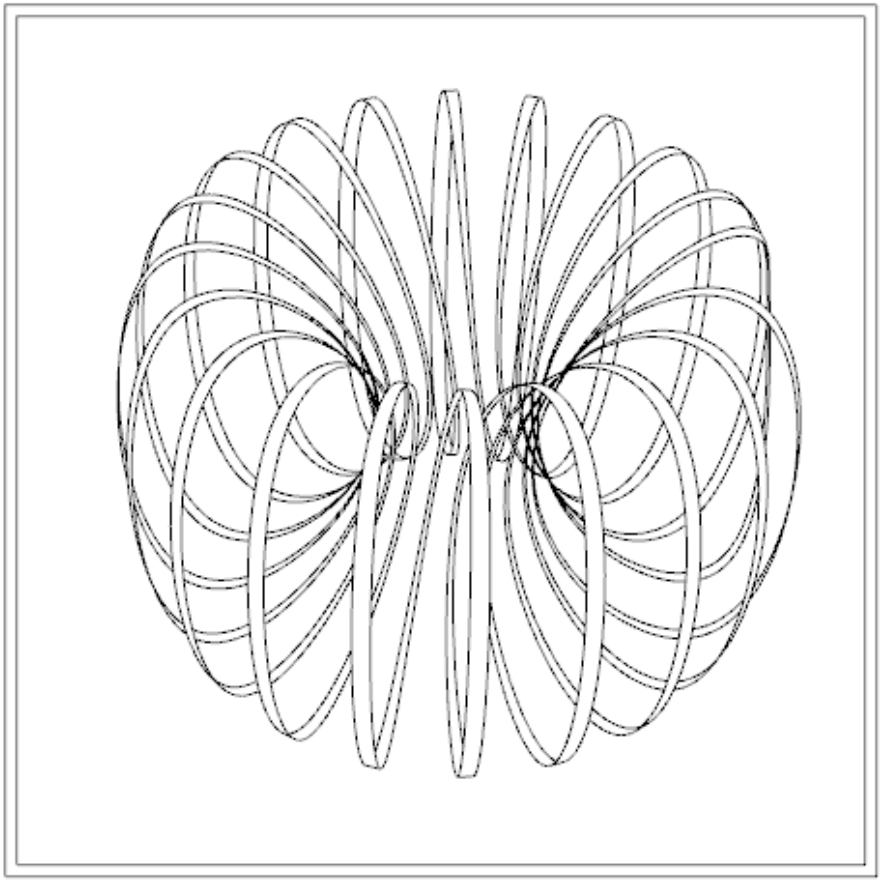
Produktio-osan aikana tuotettu materiaali

Markkula, S. (2017). *Algorithms-Aided Form-Giving Course wiki*. Wiki.aalto.fi. Saatavissa: https://wiki.aalto.fi/display/ARTSGenCAD/Arts_DOD_Algorithms_Aided_Design+Home [Viitattu 16.4.2017].

Liite 1: Tutorial 5: Slinky Doughnut

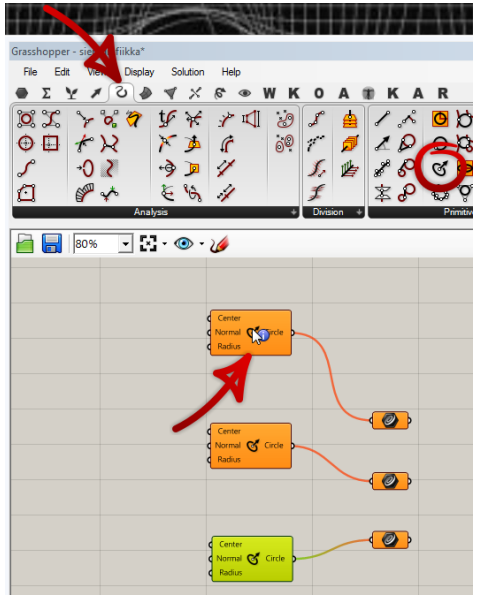
Step-by-step guide

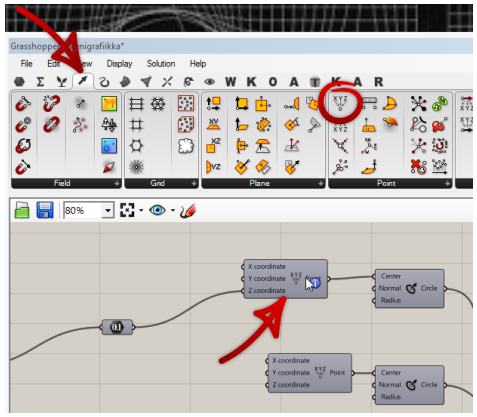
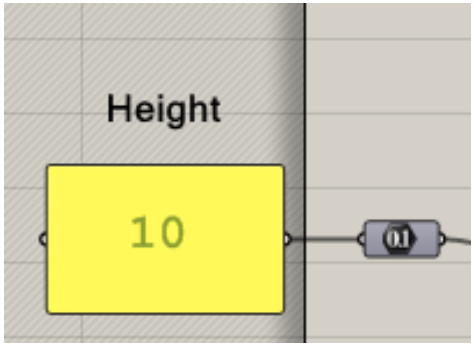
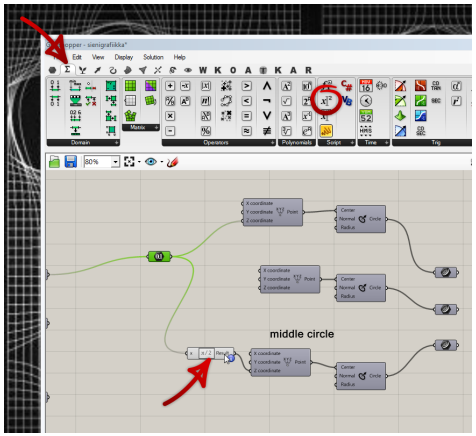
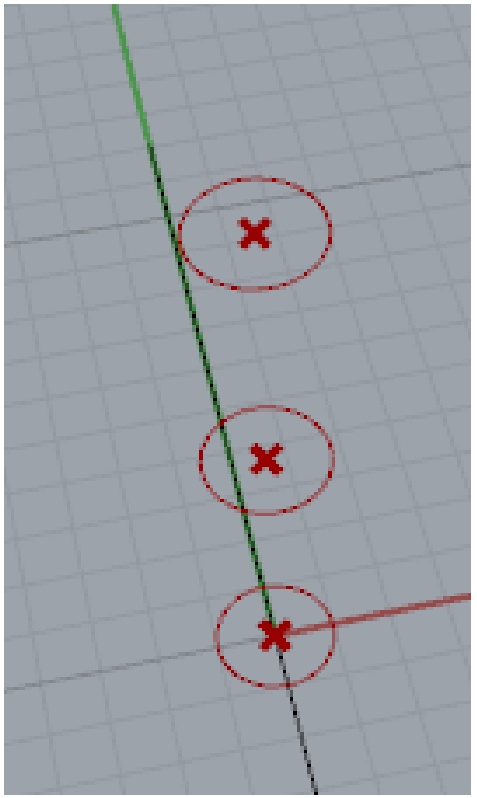
Slinky doughnut



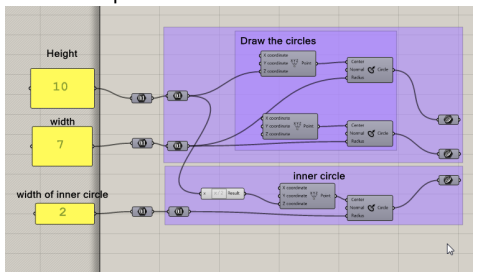
Learning points:

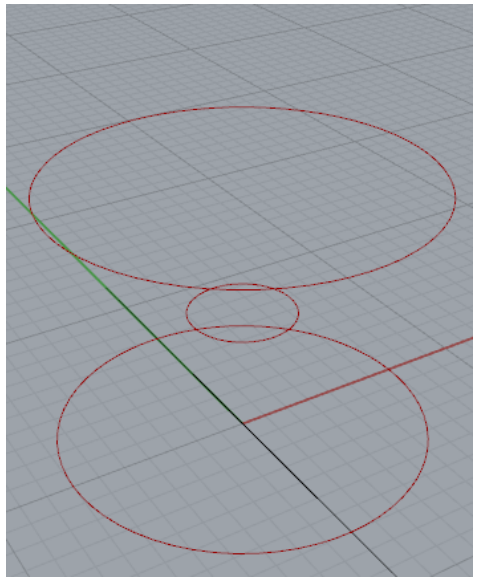
- Weave lists to one
- Interpolate curve through points
- Closed interpolate curve ends
- Using custom components
- Referencing other peoples work

Description	Component name(s) and Notes	View in rhino
To make the spiral we make points on three circles. Through these points we draw a curve. With the curve we make a surface spiraling around the doughnut		
Make 3 circular curves  	<Circle CNR>	

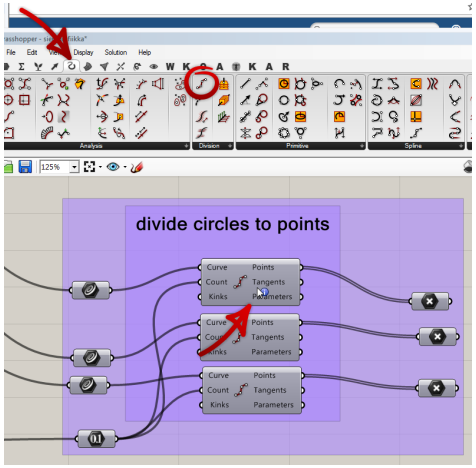
Place the one circle center to be at (0,0,0), one to the height **Height** 	<Construct Point> Input: 	
Place the third circle in middle of the two previous, by dividing the height parameter in tw, and adding this as an input to the Construct Point 	<Expression> Remove one input in the expression, by zooming in and pressing on the minus – sign. Rename the input as x. In the expression write x/2.	

Add same width to the circles on top and bottom and one to the middle circle. Group and name the funktion



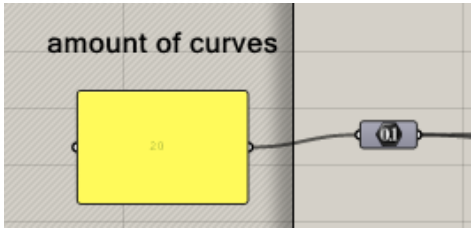


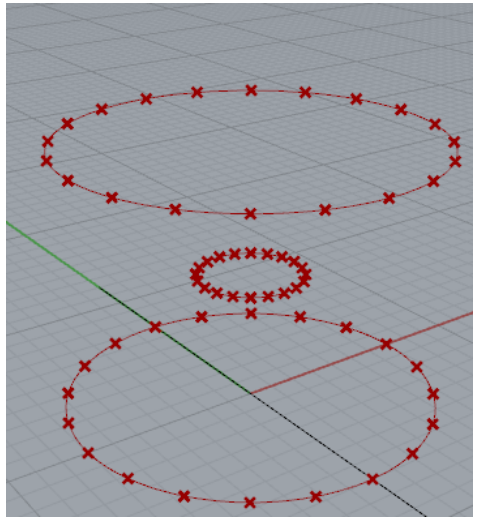
Divide the curves to get points on the circle



<Divide Curve>

Input:

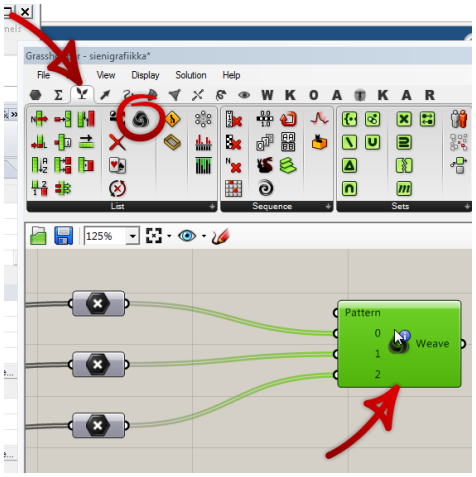




We want to make a curve that goes through point 1 in the bottom circle, point 1 in the middle circle, point 1 in top circle, point 2 in bottom circle and so on.

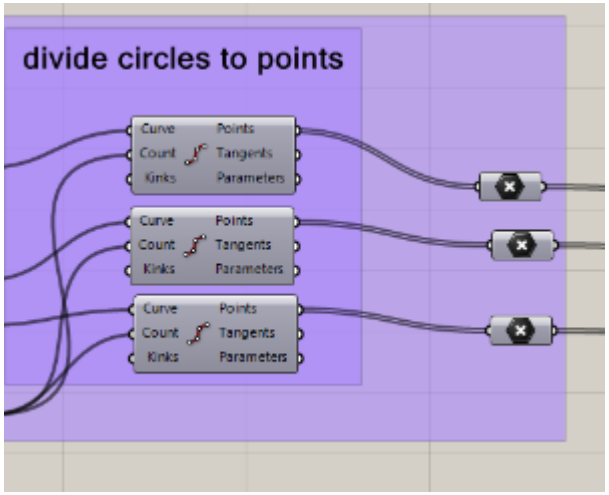
In order to do so we can weave the three lists of points into one list that has all the points in the order we want them to be. We need to add one input to a total of three.

Weave the three lists

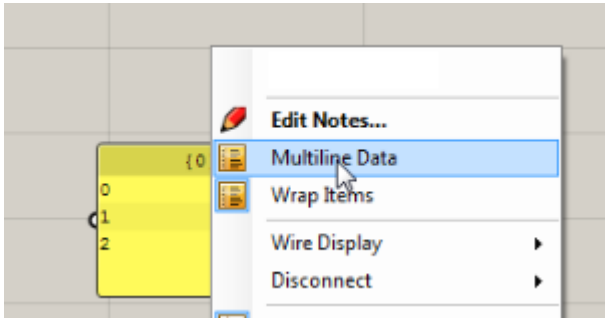


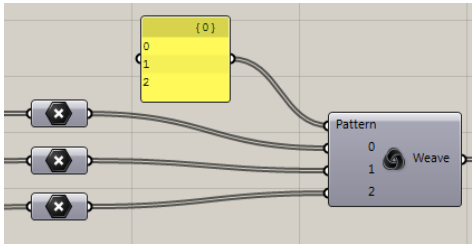
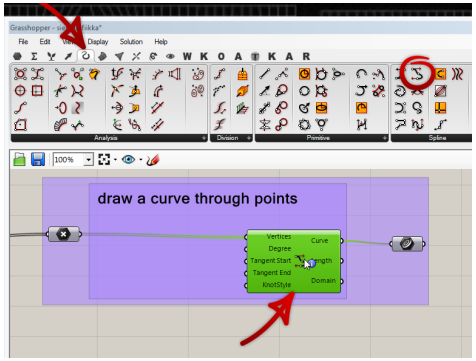
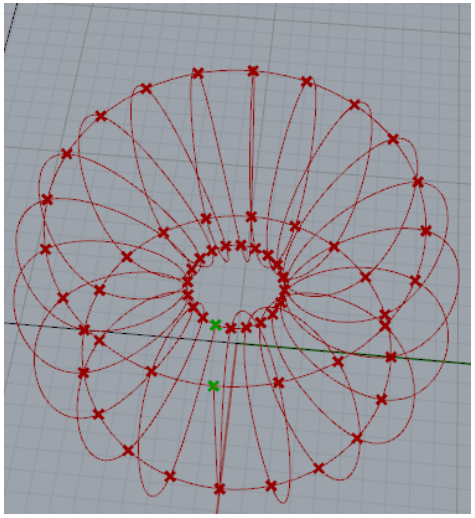
<Weave>

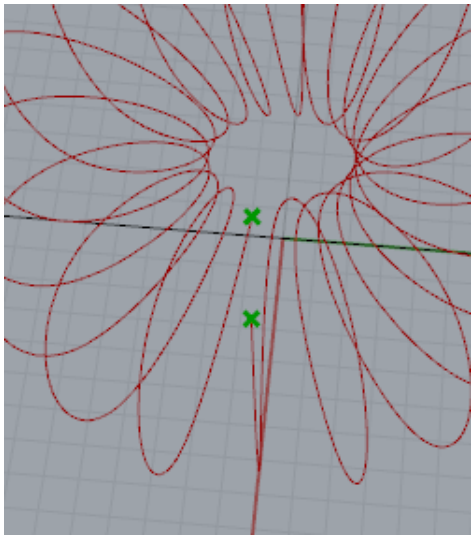
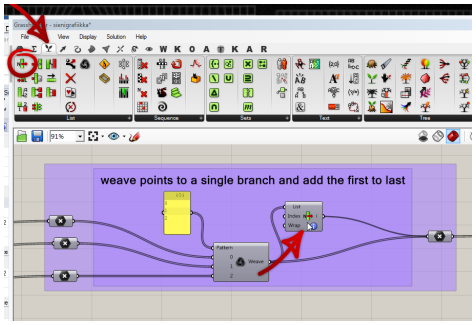
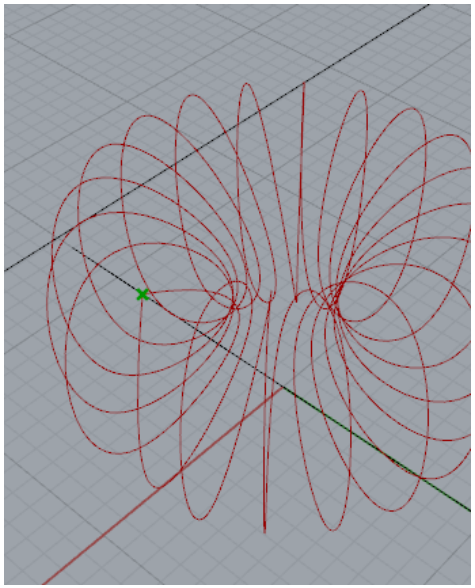
Input:

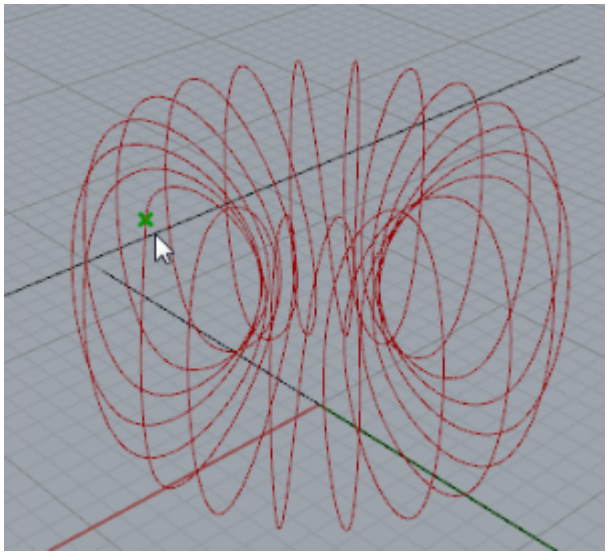


The default pattern is 0,1,0,1 and will thus not do anything with the third list.. We need to change the pattern to 0,1,2 to get all three lists weaved. We can add the pattern with a Panel. Write
0
1
2
in the panel, and right click on the panel and change "edit notes to" **Multiline Data**

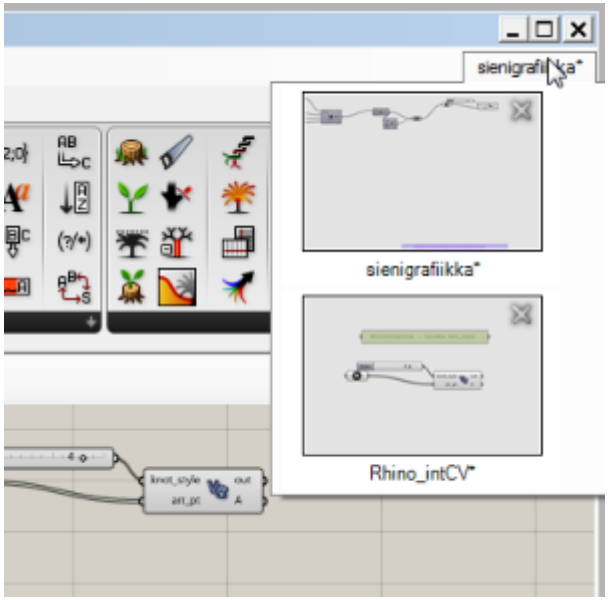


Change the pattern 		
Draw a curve through the points 	<Interpolate (t)>	

We notice, that our curve does not close (marked in green in the Rhino -picture). This is caused b the fact that we made an interpolate curve through the points, but did not make sure that we would get back to the point we started from. Let us correct for this.		
Add the first point in the list of points to be also the last point. Connect the weave-component to the Output Point first and then by holding shift add the element from the List Item. 	<List Item> The default input for List Item Index is 0, so we get the first in the list without adding more inputs then the List Item Component.	

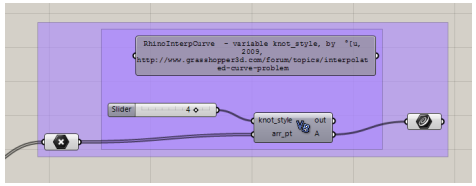
Our definition produces interesting results, but the result is still not what we want. The Grasshopper Interpolate curve does not do a great job at guessing the tangent at the end points. We can manually add the tangent to the Interpolate Curve component, but at the moment the component does not produce good closed curves. The tangency is good, but the curvature is wrong. If we know that we have an even number of points that the curve goes through the easiest way to fix the tangency is to cut the curve in half and rotate the good part 180 degrees and join the good parts. An other solution is to search online for a component or plug-in, that would do interpolate curves better.		
At http://www.grasshopper3d.com/forum/topics/interpolated-curve-problem we find a discussion regarding the same topic that we are having trouble with. There user °[u gives us a tool for solving the problem. "hi, i know your porblem... i think it is currently impossible with the grasshoppercomponents... try my vb_component... i used "RhUtil.RhinoInterpCurve" you can experiment with the knot_style (same to rhino)->the look of the curve will change (best solution with knot_style=4) good luck [u" Download 'Rhino_intCV', from grashopper3d -forum		

.ghx -files can be used by drag&dropping them to the grasshopper window. GH opens a new definition with the new component. Just copy and paste it to our definition. You can change between open definitions in the upper right by clicking on your definitions name.



Use the 'Rhino_intCV' custom vb_component

Remember to reference the the maker of the work of others, according to the model in Tutorial 3, Good practices and commenting your definition

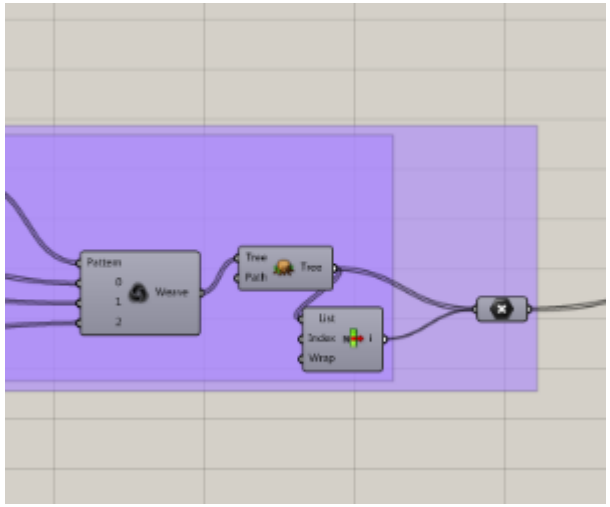


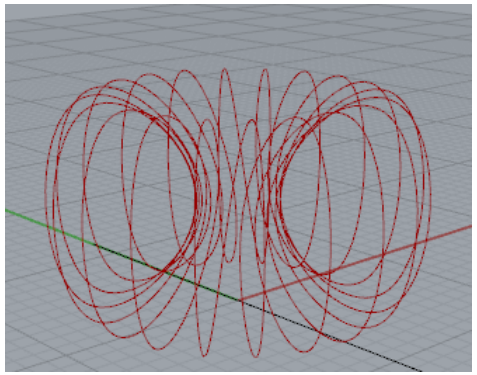
Rhino_intCV

(by °[u, 2009)

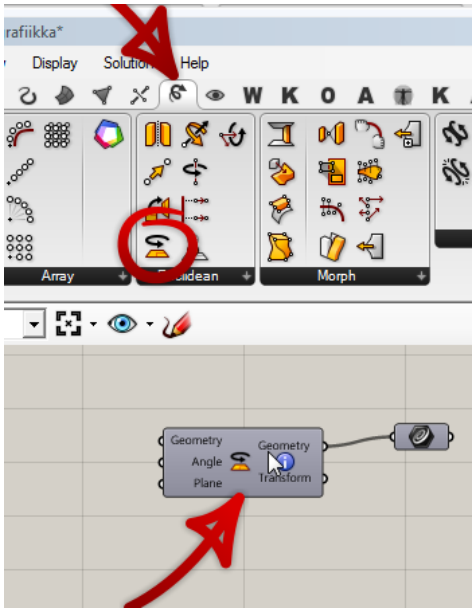
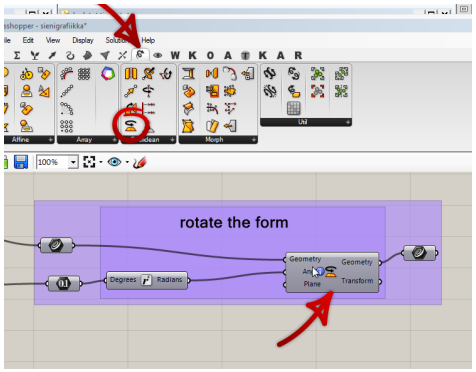
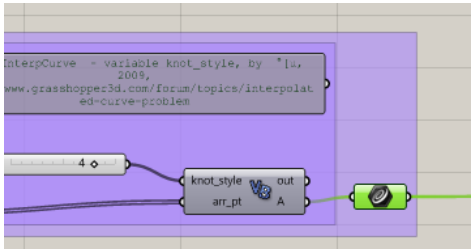
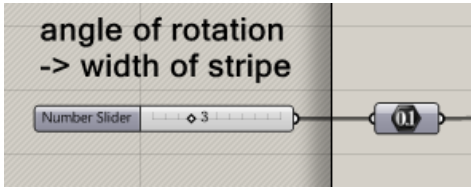
not native to Grasshopper

Input:



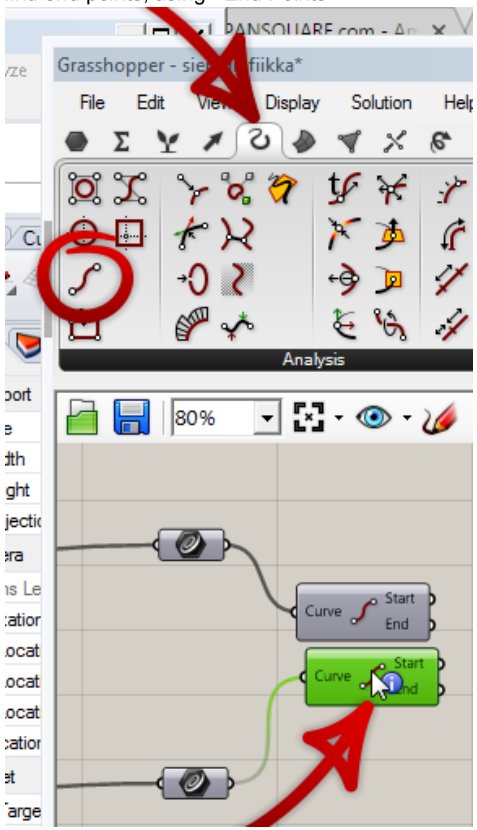


To make thickness to our curve let us rotate the whole curve and do a 2 Rail Sweep between the two spirals

Rotate the spiral curve 	<Rotate>	
Change degrees to radians to make the turn easier to understand 	<Radians> Input:  	

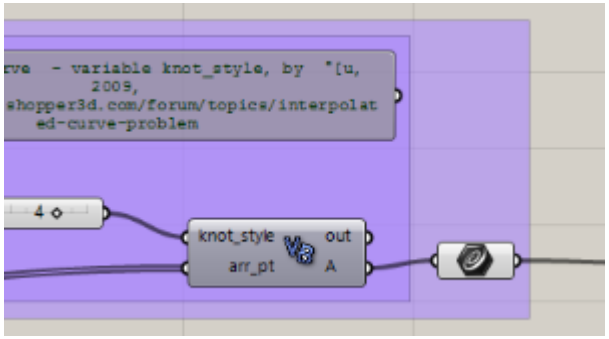
make the line to be the section for <sweep 2>.

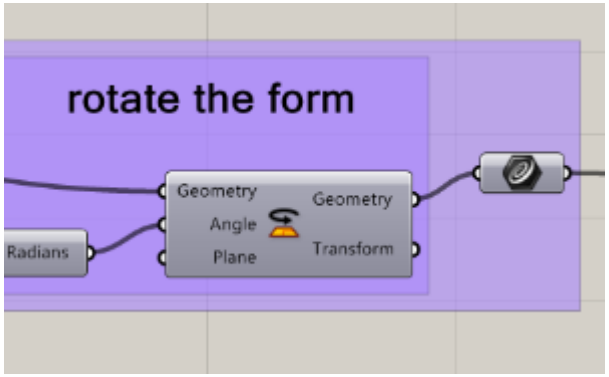
find end points, using <End Points>

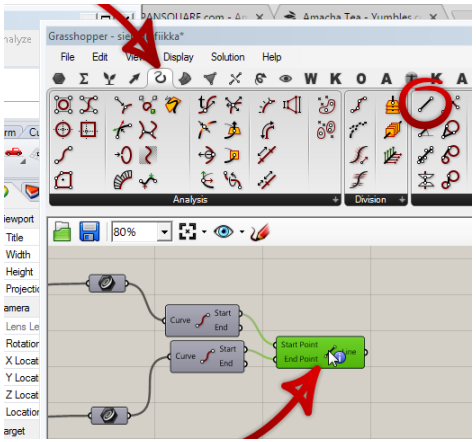
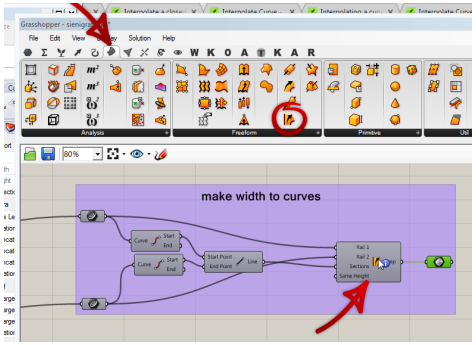
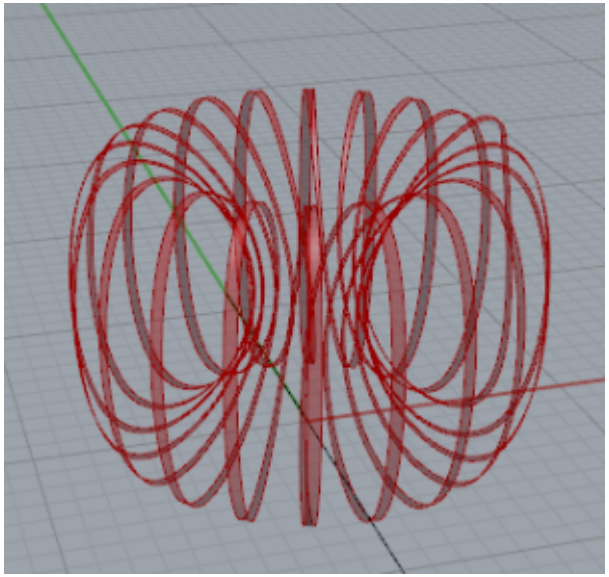


<End points>

Input:





Make a line through end points  	<Line>	
Do 2 rail sweep  	<Sweep 2>	

Bake the Brep to Rhino